

Üæŕ, æ Á&@å~ |ã \* Á^å~ &•Á@Áç] ^&cåÁ! [ b&cÁ æ^•] æ  
Á  
ŸŸY ^} åãÁãæ Áæ åÁÖ!ã ÁÖ^ { ^~ |^ { ^^•c!

# Railway scheduling reduces the expected project makespan

Wendi Tian<sup>1,2</sup> and Erik Demeulemeester<sup>1</sup>

<sup>1</sup>*Katholieke Universiteit Leuven, Research Center for Operations Management, Leuven, Belgium*

<sup>2</sup>*School of Management, Huazhong University of Science and Technology, Wuhan, China*

[wendi.tian@econ.kuleuven.be](mailto:wendi.tian@econ.kuleuven.be) and [erik.demeulemeester@econ.kuleuven.be](mailto:erik.demeulemeester@econ.kuleuven.be)

**Abstract:** The Critical Chain Scheduling and Buffer Management (CC/BM) methodology, proposed by Goldratt (1997), introduced the concepts of feeding buffers, project buffers and resource buffers as well as the roadrunner mentality. This last concept, in which activities are started as soon as possible, was introduced in order to speed up projects by taking advantage of predecessors finishing early. Later on, the railway scheduling concept of never starting activities earlier than planned was introduced as a way to increase the stability of the project, typically at the cost of an increase in the expected project makespan. In this paper, we will indicate a realistic situation in which railway scheduling improves both the stability and the expected project makespan over roadrunner scheduling.

**Key words:** *railway scheduling, roadrunner scheduling, feeding buffer, priority list, resource availability*

## 1 Introduction

The well-known resource-constrained project scheduling problem (RCPSP) is a basic problem type of the project scheduling problem that has been studied heavily for many decades (Demeulemeester and Herroelen, 2002). It involves the non-preemptive scheduling of project activities subject to finish-start, zero-lag precedence constraints and renewable resource constraints with the objective of minimizing the project duration. For this problem, both the duration of each activity and its resource requirements are usually assumed to be known, deterministic values. However, in many cases, for example in construction and software development projects, it often occurs that only one renewable bottleneck resource (e.g. labor, person-days) is available in a constant amount throughout the project. During the project planning phase, project management traditionally relies on the work breakdown structure to specify work packages and to estimate the work content (e.g. amount of person-days) for each individual activity. In

practice, several scenarios are available for the execution of the individual activities. Given an activity's work content, a set of allowable execution modes can be specified for its execution, each characterized by a fixed duration (e.g. days) and a constant resource requirement (e.g. units/day), the product of which is at least equal to the activity's specified work content.

The resulting discrete time/resource trade-off problem (DTRTP) has been introduced in De Reyck (1998), De Reyck et al. (1998) and Demeulemeester et al. (2000). The duration of an activity is assumed to be a discrete, non-increasing function of the amount of a single renewable resource committed to it. Given the specified work content  $W_i$  for activity  $i$  ( $1 \leq i \leq n$ ), all  $M_i$  efficient execution modes for its execution are determined based on time/resource trade-offs. Activity  $i$ , when performed in mode  $m$  ( $1 \leq m \leq M_i$ ), has a duration  $d_{im}$  and requires a constant amount  $r_{im}$  of the renewable resource during each period it is in progress, such that  $r_{im} * d_{im}$  is at least equal to  $W_i$ . A mode is called efficient if every other mode has either a higher duration or a higher resource requirement. Without loss of generality, we assume that the modes of each activity are sorted in the order of non-decreasing duration. The single renewable resource has a constant per period availability  $a$ . We assume that the project is represented using an activity-on-the-node network. The dummy start node 0 and the dummy end node  $n+1$  have a single execution mode with zero duration and zero resource requirement. The objective is to schedule each activity in one of its modes, subject to the finish-start precedence and the renewable resource constraints, under the objective of minimizing the project makespan. Using the classification scheme of Herroelen et al. (1999,2000), the problem is represented as  $1,1|cpm, disc, mu|C_{max}$ . The DTRTP has been shown to be strongly NP-hard (De et al., 1997).

In addition, the multi-mode resource-constrained project scheduling problem (MRCPSP) is a generalized version of the RCPSP and of the DTRTP, which involves the selection of an execution mode for each activity (mode assignment) with a specific activity duration and resource requirements and which determines

the activity start or finish times such that the project makespan is minimized and the precedence and resource constraints are satisfied. This problem is denoted as  $m,1T|cpm, disc, mu|C_{max}$  using the classification scheme of Herroelen et al. (1999). As a generalization of the RCPSP, the MRCPSP is NP-hard (Kolisch et al., 1995). Besides, Goldratt's Critical Chain Scheduling and Buffer Management (CC/BM) methodology, which is the direct application of the Theory of Constraints (TOC) to project planning and which relies on deterministic scheduling techniques in order to build a baseline schedule that is made robust by inserting various types of buffers, has attracted considerable attention and has found its way towards practical application using commercial software for scheduling the resource-constrained project scheduling problem (RCPSP) in a robust way. CC/BM removes the hidden safety in the durations of activities and aggregates these in the form of a project buffer to protect due-date performance and to avoid wasting this safety through bad multitasking, the student syndrome, Parkinson's Law and poorly synchronized integration. It believes that the critical chain is the longest chain of both precedence and resource dependent activities that determines the overall duration. Feeding buffers are placed whenever a non-critical chain activity joins the critical chain in order to protect the critical chain. During the execution, CC/BM introduces the roadrunner mentality which involves that during project execution activities may be started before their planned starting time in an effort to speed up the project by taking advantage of predecessors finishing early. During the past application of CC/BM, the implicit assumption was made that one considers only one execution mode (one duration with the corresponding resource requirements) for each activity and uses the roadrunner mentality during the execution of the project. However, little research has been performed on the application of CC/BM to the case of multiple execution modes. Therefore, we tried to apply CC/BM to the DTRTP, which is a restricted version of the MRCPSP. In the meantime, the railway scheduling policy, in which activities can never be started before their planned starting time, was also proposed as a way to increase the stability of the project, typically at the cost of an increase in the expected

project makespan. We aim to indicate a situation in which railway scheduling improves both the stability and the expected project makespan over roadrunner scheduling.

The reminder of the paper is organized as follows. The next section gives a literature review containing three parts: literature about the work content, an overview of solutions for the DTRTP including methods for solving the MRCPSPP and literature about robust project scheduling. Section 3 describes the set-up of our computational experiments. In section 4 the results of the three computational experiments are reported as well as the comparison of the four performance indicators with the roadrunner scheduling and railway scheduling policy. The last section provides our overall conclusions and offers some suggestions for future research.

## **2 Literature review**

In this paper, we consider project activities that are characterized by a work content in terms of resource-time units instead of the fixed duration and the constant resource requirements. However, the work content literature is very sparse. Elmaghraby (2005) insisted that uncertainty resides in two domains. One is “external” to the activity, such as the weather conditions, worker absenteeism and equipment failure. The other is “internal” to the activity and resides in the estimates of this work content. Traditionally, the focus has been on the former aspect with little attention to the latter. But he believed that in many projects, especially those which are more research oriented, the “internal” factor plays the dominant role. In most of the literature, it has been suggested (Demeulemeester et al. (2000), De Reyck et al. (1998), Ranjbar and Kianfar (2007) and Ranjbar et al. (2009)) that the work content translates into a set of alternative execution modes for the activities and each mode refers to a combination of a constant duration and a time invariant resource requirement. Vanhoucke and Debels (2008) investigated the influence of variable activity durations under a fixed work content, the possibility of allowing activity pre-emption and the use of fast tracking (the

compression of a project schedule by doing activities in parallel that would normally be done in a sequence) to decrease a project's duration. Fündeling and Trautmann (2010) proposed a priority rule scheduling method under work content constraints which provided a feasible resource usage profile for each activity.

The literature on the solution approach for the DTRTP is also relatively sparse. Demeulemeester et al. (2000) developed a branch-and-bound algorithm for the DTRTP based on the concept of activity-mode combinations. De Reyck et al. (1998) developed heuristic solution procedures for the DTRTP, based on local search and tabu search. Ranjbar and Kianfar (2007) developed a meta-heuristic procedure based on a genetic algorithm. Long and Ohsato (2008) analyzed the project planning and execution of this problem by using the fuzzy critical chain method. Recently, Ranjbar et al. (2009) presented a new hybrid meta-heuristic algorithm based on a scatter search and path relinking methods to solve the DTRTP with multiple renewable resources (MDTRTP).

As already mentioned, the DTRTP is a subproblem of the MRCPSP, so there are many algorithms for the general MRCPSP that can also be used to solve the DTRTP. Indeed, numerous solution procedures (both exact and heuristic) for the MRCPSP have been proposed in the past three decades. Chapter 8 of the project scheduling research handbook of Demeulemeester and Herroelen (2002) and Van Peteghem and Vanhoucke (2010) give a literature review of procedures for the MRCPSP. On the basis of their work, an overview of procedures for the MRCPSP is shown in table 1.

Table 1: Literature review for the *MRCPSP*

Author(s)	Year	Method	Dataset	Number of activities	$ M $	R/N R	R	NR
Slowinski	1980	LP	own	-	-	RNR	-	-
Talbot	1982	B&B	own	10,20,30	1-3	RNR	3	0
Patterson et al.	1989	B&B	-	30	3	RNR	-	-
Speranza & Vercellis	1993	B&B	own	10-20	$\geq 2$	RNR	1-6	1
Boctor	1993	Heur	own	50,100	1-4	R	1,2,4	0
Drexler & Grünwald	1993	Heur	own	10	2-4	RNR	3	1
				10	2-4	RNR	3	3
Sprecher	1994	B&B	own	10	3	RNR	2	2
Özdamar &	1994	Heur	own	20-57	1-3	RNR	1-6	1-6

Ulusoy								
Slowinski et al.	1994	SA	own	30	2	RNR	3	3
Boctor	1996 a	SA	Boctor (1993)	50,100	1-4	R	1,2,4	0
Boctor	1996 b	Heur	Boctor (1993)	50,100	1-4	R	1,2,4	0
Hartmann & Sprecher	1996	Speranza & Vercellis (1993)	-	-	-	RNR	-	-
Sprecher et.al.	1997	B&B	PSPLIB	10	3	RNR	2	2
Sung & Lim	1997	Heur	own	20,30,50	2-4	R	4	0
Mori & Tseng	1997	GA	own	20,30,40, 50,60,70	2-4	R	4	0
Kolisch & Drexl	1997	Heur	PSPLIB	10, 30	3	RNR	2	2
Hartmann & Drexl	1998	B&B	PSPLIB	10,12,14,16	3	RNR	2	2
Sprecher & Drexl	1998	B&B	PSPLIB	10,12,14, 16,18,20	3,1-5	RNR	2,1-5	2, 1-3
			own	10,12,14, 16,18,20	3	R	2	0
Özdamar	1999	GA	PSPLIB	10	2	RNR	3	0
			own	90	2	RNR	2	2
Knotts et al.	2000	Heur	Maroto & Tormos (1994)	50	2	R	3	0
Nonobe & Ibaraki	2001	TS	PSPLIB	30	3	R	2	2
Józefowska et al.	2001	SA	PSPLIB	10,12,14, 16,18,20,30	3	RNR	2	2
Hartmann	2001	GA	PSPLIB	10,12,14, 16,18,20,30	3	RNR	2	2
Bouleimen &Lecocq	2003	SA	PSPLIB	10,12,14, 16,18,20,30	3	RNR	2	2
Alcaraz et al.	2003	GA	PSPLIB	10,12,14, 16,18,20,30	3	RNR	2	2
			Boctor (1993)	50,100	1-4	NR	1,2,4	0
Zhang et al.	2006	PSO	PSPLIB	10,12,14, 16,18,20	3	RNR	2	2
Zhu et al.	2006	B&C	PSPLIB	20, 30	3	RNR	2	2
Lova et al.	2006	Heur	Boctor (1993)	50,100	1-4	R	1,2,4	0
Jarboui et al.	2008	CPSO	PSPLIB	10,12,14, 16,18,20,30	3	RNR	2	2
Ranjbar et al.	2009	HSS	PSPLIB	10,12,14, 16,18,20	3	RNR	2	2
Lova et al.	2009	GA	PSPLIB	10,12,14, 16,18,20,30	3	RNR	2	2
			Boctor (1993)	50,100	1-4	NR	1,2,4	0
Van Peteghem & Vanhoecke	2010	BPGA	PSPLIB	10,15,20, 25,30	3	RNR	2	2
			Boctor (1993)	50,100	1-4	NR	1,2,4	0

**Remark:** R/NR = applicable on datasets with renewable resource (R) or with both renewable and nonrenewable (RNR )

resources;  $|M|$  = number of modes; R = number of renewable resource; NR = number of nonrenewable resources.

### Exact algorithms

Slowinski (1980) was the first to present a one-stage and two-stage linear programming approach to solve the multi-mode problem. Talbot (1982) introduced a 0-1 programming model and presented an enumeration scheme based procedure. Later on, there are a lot of enumeration scheme based procedures presented in the literature which can be classified into three basic categories:

- (1) the precedence tree scheme: Patterson et al. (1989) refined Talbot's procedure by introducing an enumerative type of branch-and-bound algorithm based on the generation of a precedence tree that guides the search for solutions. Sprecher (1994) improved Talbot's procedure by adding dominance and bounding rules. Speranza and Vercellis (1993) proposed a depth-first branch-and-bound algorithm, but Hartmann and Sprecher (1996) have shown that this algorithm is flawed because in some cases it is unable to determine the optimal solution. Sprecher and Drexl (1998) developed a new procedure based on the precedence tree and improved it by including new bounding criteria.
- (2) the scheme based on mode and delay alternatives: Sprecher et al. (1997) extended the concept of delaying alternatives introduced by Demeulemeester and Herroelen (1992) for the single mode RCPSP and defined the concepts of delay and mode alternatives.
- (3) the scheme based on mode and extension alternatives: Hartmann and Drexl (1998) proposed an alternative exact approach based on the concepts of mode and extension alternatives to solve the *MRCPS*P and compared this method with the other two enumeration schemes.

Besides, Zhu et al. (2006) proposed a branch-and-cut algorithm which is a generalization of branch-and-bound to solve the *MRCPS*P. All these exact algorithms perform well for small sized projects but are unable to find an optimal solution for larger problems in a reasonable computation time. So, many heuristic procedures are proposed for specific versions of multi-mode project scheduling problems.



### Heuristic procedures

Talbot (1982) recommended the use of a truncated version of his exact enumeration procedure. Boctor (1993) compared 21 heuristic scheduling rules to identify the most efficient heuristics and suggested a combination of five heuristics which have a high probability of giving the best solution. Drexl and Grünewald (1993) proposed a regret-based biased random sampling approach based on the joint use of a serial scheduling scheme and the shortest processing time rule. Özdamar and Ulusoy (1994) proposed a local constraint based analysis approach. Boctor (1996b) presented a heuristic algorithm based on the critical path method computation. Kolisch and Drexl (1997) presented a local search method with a single neighborhood search. Sung and Lim (1997) developed a branch-and-bound procedure using two lower bounds, which are incorporated in a two-phase heuristic method. Knotts et al. (2000) evaluated different agent based algorithms. Lova et al. (2006) designed several multi-pass heuristics based on priority rules for solving the *MRCPS*.

In addition, there are some meta-heuristics presented for the *MRCPS*. Slowinski et al. (1994), Boctor (1996a), Józefowska et al. (2001) and Bouleimen and Lecocq (2003) used the simulated annealing approach, while Mori and Tseng (1997), Özdamar (1999), Hartmann (2001) and Alcaraz et al. (2003) presented genetic algorithms. Lova et al. (2009) developed a hybrid genetic algorithm (HGA) which used a powerful local search method to improve the solutions provided by GA to solve the *MRCPS*. Van Peteghem and Vanhoucke (2010) presented a bi-population genetic algorithm (BPGA) to the *MRCPS* and its preemptive version. Nonobe and Ibaraki (2001) proposed a tabu search procedure and Zhang et al. (2006) presented a particle swarm optimization approach for solving the *MRCPS*. Jarboui et al. (2008) used a combinatorial particle swarm optimization (CPSO) algorithm to tackle the *MRCPS* and compared it with a simulated annealing and particle swarm optimization algorithm. Ranjbar et al. (2009) presented a hybrid scatter search and used the path relinking methodology as a solution combination method.

During project execution, there may occur some disruptions due to uncertainty. Research in project scheduling has focused on proactive and reactive procedures to counteract the effects of these disruptions as much as possible: proactive planning attempts to build a stable project plan that protects against the possible disruptions that may occur during project execution, while the reactive planning procedures are called every time the disruption changes the baseline schedule such that it cannot be executed anymore as planned. Herroelen and Leus (2004) gave a review and classification of procedures for the robust and reactive project scheduling. Van de Vonder et al. (2005) investigated the potential trade-off between the stability and the makespan by applying the critical chain logic and by using feeding and project buffers for the *RCPSP*.

### 3 Computational set-up

In this section, in order to describe the computational set-up clearly, we assume that projects are presented in activity-on-the-node representation, where the precedence constraints are of the finish-start type with a zero time-lag. An example network with ten real activities is given in Fig. 1: nodes 0 and 11 are the dummy start and end activities, respectively. There is only one renewable resource type with 10 available units in each time period. The number above the node represents the mean work content which is obtained randomly in a way that will be described later on in this section.

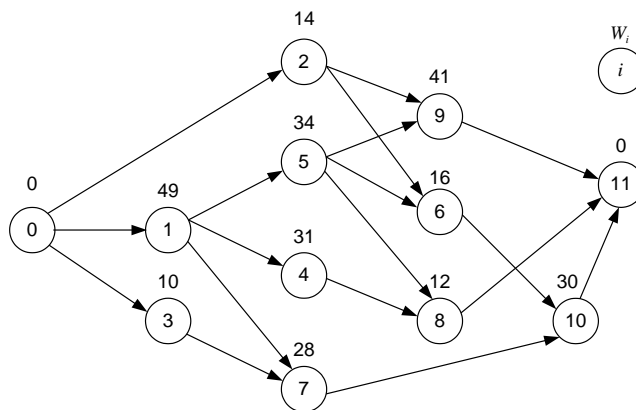


Fig. 1 An example network

Two execution policies and four performance indicators are proposed in sections 3.1 and 3.2 respectively. Then in sections 3.3 and 3.4, we introduce the way to get possible modes and obtain all optimal mode combinations. Three experiments are designed and explained in detail in section 3.5. We describe the test set in section 3.6.

### 3.1 Two execution policies

Two execution policies are considered in this paper. One is the roadrunner scheduling policy and the other one is the railway scheduling policy:

- (1) **Roadrunner scheduling:** it is also named relay racer behavior or roadrunner mentality which is typically applied during the execution of a *CC/BM* schedule. The activities of the project should start as soon as possible when all their predecessors have finished and when enough resource units are available to start. This policy is executed in the hope of decreasing the expected project length.
- (2) **Railway scheduling:** this policy is derived from the train schedule. Trains are scheduled to leave a railway station at a certain time and are not allowed to do so at an earlier time. So railway scheduling means that every activity of the project should not start earlier than its planned starting time. This policy is mainly introduced in order to increase the stability of the project.

### 3.2 Four performance indicators

In order to compare the performance of different applications of the roadrunner scheduling and railway scheduling policies, we introduced four performance indicators as follows:

- (1) **Average project length (APL):** the average completion time of finishing the whole project while satisfying all the resource and precedence constraints.

The function is  $APL = \frac{\sum_{t=1}^N S_{nt}}{N}$ , in which  $S_{nt}$  denotes the start (and thus the

completion) time of the dummy end activity of the project in simulation run  $t$  and  $N$  represents the number of simulation runs. The smaller the average

project length, the better.

- (2) **Standard deviation of the project length (SDPL)**: the variability of the

project length and the function is  $SDPL = \sqrt{\frac{\sum_{t=1}^N (S_{nt} - APL)^2}{N-1}}$ . The smaller

the standard deviation of the project length, the better.

- (3) **Timely project completion probability (TPCP)**: the probability of the project finishing within the due date. The function is

$TPCP = P(S_{nt} \leq \text{duedate})$ . The larger the timely project completion probability, the better. In this paper, the due date is set 20% above the optimal makespan of the deterministic case.

- (4) **Stability cost (SC)**: the penalty cost of deviating from the planned starting

times. The stability cost function is  $SC = \frac{\sum_{t=1}^N \sum_{i=1}^n w_i |S_{it} - s_i|}{N}$ , which means

the weighted sum of the expected absolute deviations between the planned starting time  $s_i$  of activity  $i$  and its realized starting time  $S_{it}$  during simulation run  $t$  where  $w_i$  is the penalty weight. In this paper, we use Van de Vonder et al.'s (2008) weighting function. It belongs to a triangular distribution with  $P(w_i = q) = (21 - 2q)\%$ ,  $q \in \{1, 2, \dots, 10\}$ . The distribution results in a higher probability for low weights and in an average weight  $w_{ave} = 3.85$ . The weight of the dummy end activity denotes the marginal cost of not completing the project within the due date and is set to 38. Remark that finishing the project early or at the due date results in no extra stability cost. The smaller the stability cost, the better.

### 3.3 Generation of all possible execution modes

The mean work content of every activity and its standard deviation were randomly generated from the uniform distribution between 10 and 50 and 1 and 5, respectively. The mode generation in this paper is slightly different from the mode generation by De Reyck et al. (1998). In their paper, the authors generated the first

mode for activity  $i$  with duration  $d_{i1} = \max\{\lfloor \sqrt{W_i} \rfloor, \lceil W_i / a \rceil\}$  and resource requirement  $r_{i1} = \lceil W_i / d_{i1} \rceil$ , where  $\lfloor x \rfloor$  denotes the highest integer number equal to or smaller than  $x$  and where  $\lceil x \rceil$  denotes the smallest integer number equal to or larger than  $x$ . Then the procedure generates a mode with duration  $d_{i1} - 1$  and corresponding resource requirement. Consequently, the mode with duration  $d_{i1} + 1$  is generated. This mode generation process continues (alternatively decreasing, resp. increasing the mode duration) until the desired number of modes is reached or no more modes are left. In this paper, we generate the resource requirements  $r_{im}$  from  $a$  to 1, correspondingly calculate the duration  $d_{im} = \lceil W_i / r_{im} \rceil$  and check whether the resulting mode is efficient. The mean work content, the standard deviation of the work content, the mode information and the relationship between the activities for the example in Fig. 1 are shown in table 2.

Table 2: All possible modes and relationships between the activities of the example project

Act	Work content	Standard deviation	Modes <duration, resource requirements>	$ M $	Predecessors
0	0	0.000000	{<0,0>}	1	-
1	49	2.782769	{<5,10>,<6,9>,<7,7>,<9,6>,<10,5>,<13,4>,<17,3>,<25,2>,<49,1>}	9	0
2	14	1.018677	{<2,7>,<3,5>,<4,4>,<5,3>,<7,2>,<14,1>}	6	0
3	10	2.511521	{<1,10>,<2,5>,<3,4>,<4,3>,<5,2>,<10,1>}	6	0
4	31	3.284738	{<4,8>,<5,7>,<6,6>,<7,5>,<8,4>,<11,3>,<16,2>,<31,1>}	8	1
5	34	3.428663	{<4,9>,<5,7>,<6,6>,<7,5>,<9,4>,<12,3>,<17,2>,<34,1>}	8	1
6	16	3.652181	{<2,8>,<3,6>,<4,4>,<6,3>,<8,2>,<16,1>}	6	2,5
7	28	2.408490	{<3,10>,<4,7>,<5,6>,<6,5>,<7,4>,<10,3>,<14,2>,<28,1>}	8	1,3
8	12	3.430738	{<2,6>,<3,4>,<4,3>,<6,2>,<12,1>}	5	4,5
9	41	4.210425	{<5,9>,<6,7>,<7,6>,<9,5>,<11,4>,<14,3>,<21,2>,<41,1>}	8	2,5
10	30	2.207801	{<3,10>,<4,8>,<5,6>,<6,5>,<8,4>,<10,3>,<15,2>,<30,1>}	8	6,7
11	0	0.000000	{<0,0>}	1	8,9,10

### 3.4 Obtain all optimal baseline schedule (OBS)

There are millions of combinations of efficient modes for this problem, and the complexity of this problem is  $O(|M|^n)$ , where  $|M|$  denotes the maximum number

of efficient modes that can be assigned to each activity and  $n$  denotes the number of activities in the project. Because of the long computation time needed for enumerating all efficient modes for each activity, we enumerated all feasible mode selections that result in a minimal makespan. Furthermore, a project manager indeed will typically select baseline schedules with a makespan that is as short as possible. So we try to obtain all the optimal schedules of these efficient mode selections, which will be used in the further experiments.

From the previous literature review, we know that there are a lot of exact and heuristic procedures available for obtaining good schedules for this problem. However, all the optimal schedules cannot be obtained by using the heuristic or meta-heuristic procedures. So we have to use an exact algorithm to obtain all the optimal baseline schedules with minimum makespan. In this paper, a slightly adapted version of the branch-and-bound procedure that was developed by Demeulemeester et al. (2000) for the *DTRTP* was used to obtain all the optimal schedules, and the remark has to be made that different schedules with the same combination of efficient modes are not considered. That is to say, although there might be many different schedules which have the same minimum makespan with one specific combination of modes, we just take one early start optimal schedule as our baseline schedule. For the example in Fig. 1, we show all the optimal mode combinations in table 3 and the corresponding optimal early start baseline schedules in table 4.

Table 3: All optimal mode combinations for the example in Fig. 1

Choice	Modes chosen	Makespan
1	<0,0>,<5,10>,<7,2>,<5,2>,<8,4>,<9,4>,<4,4>,<7,4>,<3,4>,<7,6>,<3,10>,<0,0>	27
2	<0,0>,<5,10>,<2,7>,<1,10>,<11,3>,<5,7>,<4,4>,<4,7>,<3,4>,<7,6>,<3,10>,<0,0>	27
3	<0,0>,<5,10>,<7,2>,<1,10>,<11,3>,<7,5>,<4,4>,<4,7>,<3,4>,<7,6>,<3,10>,<0,0>	27
4	<0,0>,<5,10>,<7,2>,<5,2>,<4,8>,<6,6>,<8,2>,<14,2>,<2,6>,<7,6>,<3,10>,<0,0>	27
5	<0,0>,<5,10>,<5,3>,<10,1>,<8,4>,<5,7>,<8,2>,<4,7>,<2,6>,<14,3>,<3,10>,<0,0>	27
6	<0,0>,<5,10>,<2,7>,<1,10>,<11,3>,<5,7>,<4,4>,<4,7>,<6,2>,<14,3>,<6,5>,<0,0>	27
7	<0,0>,<5,10>,<7,2>,<1,10>,<11,3>,<7,5>,<4,4>,<4,7>,<6,2>,<14,3>,<6,5>,<0,0>	27

Table 4: The optimal starting times for the mode combinations of table 3

Choice	Act 0	Act 1	Act 2	Act 3	Act 4	Act 5	Act 6	Act 7	Act 8	Act 9	Act 10	Act 11
1	0	0	10	5	5	5	20	13	14	17	24	27
2	0	0	6	5	6	8	20	13	17	17	24	27
3	0	0	10	5	6	10	20	6	17	17	24	27
4	0	0	9	5	5	9	16	10	15	17	24	27
5	0	0	5	10	10	5	10	20	18	10	24	27
6	0	0	6	5	6	8	13	17	21	13	21	27
7	0	0	6	5	6	6	13	17	21	13	21	27

### 3.5 Three experiments

Once we obtained all the optimal baseline schedules with different mode combinations, we will perform three computational experiments as follows. The first one is to compare the performances of roadrunner scheduling and railway scheduling when different feeding buffer sizes are inserted. The next experiment is to see the difference between roadrunner scheduling behavior and railway scheduling behavior when performing the simulation runs according to eight different priority lists. The impact of the resource availability is investigated in the third experiment. In these three experiments, each schedule for each scheduling policy is simulated one thousand times as these results proved to be adequate and stable. All these experiments have been coded and compiled in Microsoft Visual C++ 6.0 and were run on a Dell PC with Pentium *R* with 3.2 GHz processor and 0.99 GB of RAM.

#### 3.5.1 Experiment 1: The influence of feeding buffers

Since the publication of Goldratt's novel *Critical Chain* in 1997, CC/BM has attracted a lot of attention and was widely studied. Quite some books (Newbold (1998), Leach (2000), among others) and articles (Leach (1999), Umble and Umble (2000), Herroelen and Leus (2001), among others) have been written on this topic. The merits and pitfalls of the *CC/BM* scheduling approach have been summarized by Herroelen and Leus (2001). The critical chain is defined as a project length determining chain that not only considers the precedence relations but also takes the resource dependencies into account. Buffer management as one of the main features of *CC/BM* consists of shifting the safety time of the activities to the end of the critical chain in the form of a project buffer, and to protect the

critical chain in the form of feeding buffers that are located in places where a non-critical chain activity joins the critical chain. In the literature, there exists some research about the buffer sizing approach. Goldratt suggested all the activities to have 50% buffers as the safety time, regardless of their uncertainty levels. Newbold (1998) computed the feeding buffers and the project buffer by the Cut & Paste method and the root square error method. Tukel et al. (2006) introduced two new methods for the feeding buffer sizes. One was incorporating resource tightness and the other one was using network complexity. They compared their new methods with the Cut & Paste method and the root square error method, as well as using no buffer as a benchmark. The two new methods proved to generate smaller buffer sizes and provided sufficient protection against delays in the project completion time. However, during the application of *CC/BM* the implicit assumption is made that one considers only one execution mode with a specific duration and specific resource requirements for each activity. In this paper, we propose to apply the *CC/BM* methodology to the multiple modes problem and to compare the performances of using a roadrunner scheduling policy and a railway scheduling policy for every obtained optimal baseline schedule. In our experiment, we set the feeding buffer sizes to 0%, 10%, 20%, 30%, 40% and 50% of the corresponding non-critical chain length which can also help us to see what appropriate buffer sizes are. For the *DTRTP*, we notice that the scheme of the optimal baseline schedule is very condense and there might exist more than one critical chain. In the book by Goldratt (1997) and in other papers (Herroelen & Leus (2001), Tukel et al. (2006) and Van de Vonder et al. (2005)), typically an arbitrary choice is made if there is more than one critical chain. However, every critical chain is considered in this paper. After inserting the feeding buffers, there might be resource or precedence conflicts. Then, we have to reschedule the optimal baseline schedules. That means that the critical chain might be broken. So when we reschedule, precedence constraints between the activities on the critical chain were added to the problem in order to force this critical chain to remain a chain and the optimal procedure of Demeulemeester and Herroelen (1992) for the



RCPSP is used to construct an optimal schedule. The simulation runs were done according to these optimal rescheduled schedules (*ORS*) and obeyed the critical chain priority rule: the activity on the critical chain has priority over an activity on a non-critical chain when both are available to start at the same time.

For the example in Fig. 1, we obtained the optimal baseline schedule for the sixth choice of the mode combination (see table 3) that is shown in Fig. 2. We can see that there are five critical chains for this schedule:  $\langle 0-1-3-2-5-9-11 \rangle$ ,  $\langle 0-1-3-2-5-6-7-10-11 \rangle$ ,  $\langle 0-1-3-2-5-6-7-8-11 \rangle$ ,  $\langle 0-1-3-4-7-8-11 \rangle$  and  $\langle 0-1-3-4-7-10-11 \rangle$ . The *ORS* after inserting the feeding buffers can be seen in Fig. 3 when choosing the first critical chain. The buffer size in Fig. 3 is 50% of the non-critical chain length (rounded up to the next integer).

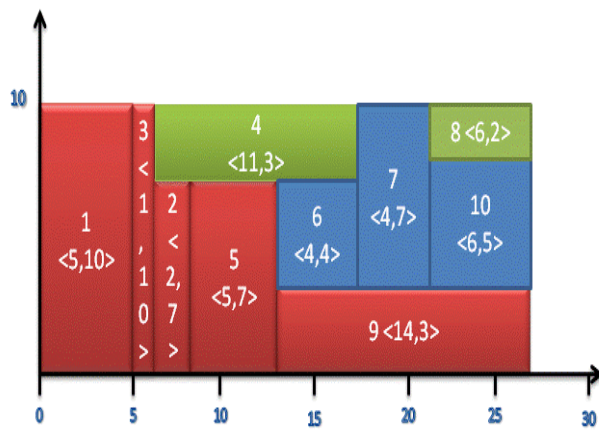


Fig. 2 OBS for the sixth choice of mode combinations

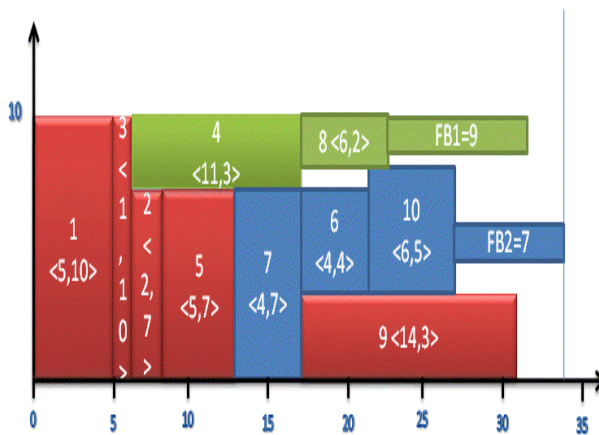


Fig. 3 ORS for the sixth choice of mode combinations

### 3.5.2 Experiment 2: Different priority lists

In this second experiment, we perform the simulation runs based on eight different priority lists. The aim of this experiment is to compare the performance of combining a roadrunner scheduling policy and a railway scheduling policy with these eight different priority lists. For every priority list, the dummy start activity is the first activity in the priority list and the dummy end activity is the last activity in the priority list. Eight different priority lists are explained in detail as follows (each time the smallest activity number is used as a tie-breaker):

(1) random activity

The priority list is sorted randomly. For every execution, a purely random priority list is generated.

(2) starting time

The priority list is sorted according to the starting time of the *OBS*. For the *OBS* in Fig. 2, the starting time priority list is <0-1-3-2-4-5-6-9-7-8-10-11>.

(3) first critical chain

The first critical chain priority list first sorts the activities on the critical chain according to smallest starting time and then the remaining activities according to smallest starting time of *ORS*. From Fig. 3, we get the priority list <0-1-3-2-5-9-4-7-6-8-10-11>.

(4) second critical chain

The second critical chain priority list is obtained by using the same method as the first critical chain priority list. The only difference is that we might get a different *ORS*. In the first critical chain priority list, we get the *ORS* through adding precedence constraints for the activities on the critical chain and rescheduling the project. However, in the second critical chain priority list, we don't add precedence constraints for the activities on the critical chain. Thus, some activities on the critical chain might overlap in the resulting *ORS*. In this case, the priority list is <0-1-3-2-5-9-4-6-7-8-10-11>.

(5) standard deviation of the estimated activity duration from small to large

The priority list is according to smallest standard deviation of the estimated

activity duration. The priority list is <0-1-2-3-7-10-5-6-4-9-8-11> (see table 5, where for each activity the average duration and standard deviation of the duration is indicated for the chosen mode as well as the ratio of these two values, which is required for priority rules 7 and 8).

(6) standard deviation of the estimated activity duration from large to small

The priority list is according to largest standard deviation of the estimated activity duration. The priority list is <0-8-9-4-6-5-10-7-3-2-1-11> (see table 5).

(7) ratio of average and standard deviation of the duration from small to large

The priority list is according to increasing ratio of average duration and standard deviation of the duration. For the example in Fig. 1 and for the sixth choice of mode combination, the priority list is <0-3-8-6-2-7-4-5-9-1-10-11> (see table 5).

(8) ratio of average and standard deviation of the duration from large to small

The priority list is according to decreasing ratio of average duration and standard deviation of the duration. For the example in Fig. 1 and for the sixth choice of mode combination, the priority list is <0-10-1-9-5-4-7-2-6-8-3-11> (see table 5).

Table 5: data for the last four priority lists based on the sixth choice of mode combinations

Act	Work content	Standard deviation of the work content	Resource requirement of optimal mode	Corresponding average duration	Standard deviation of the duration	Ratio of average duration and standard deviation
0	0	0.000000	0	0	0	-
1	49	2.782769	10	5.277	0.45	11.727
2	14	1.018677	7	2.326	0.469	4.9597
3	10	2.511521	10	1.43	0.4953	2.887
4	31	3.284738	3	10.67	1.1645	9.1634
5	34	3.428663	7	5.317	0.5664	9.387
6	16	3.652181	4	4.37	0.9592	4.556
7	28	2.408490	7	4.402	0.5026	8.7577
8	12	3.430738	2	6.3	1.7248	3.6526
9	41	4.210425	3	14.01	1.4174	9.8823
10	30	2.207801	5	6.412	0.5501	11.657
11	0	0.000000	0	0	0	-

### 3.5.3 Experiment 3: The impact of the resource availability

In this experiment, experiments 1 and 2 will be repeated three times, namely with

resource availabilities of 10, 15 and 20. Only the average results will be shown on the figures.

### 3.6 Test set

As a test set for assessing the two execution policies described in this paper, we use the well-known PSPLIB set of project network instances (Kolisch & Sprecher, 1996). As the computation times were quite large, we randomly chose 100 instances from the set of multi-mode resource-constrained project scheduling problems with 10 activities. The mean work content and the standard deviation of the work content for each instance were obtained as mentioned above. When performing the simulation runs for each schedule, we assume that the work content of every activity  $i$  belongs to a normal distribution with mean  $\mu_i$  and standard deviation  $\delta_i$ . If  $\delta_i$  is very large and  $\mu_i$  is fairly small, a negative work content might be generated. In the real world, this is impossible and meaningless. So, during the simulation, whenever the obtained work content from this normal distribution is negative, it is set to zero and thus the activity does not have to be executed (this happened 360 times in total in our experiments, which means that it appears 3.6 times out of 1000 simulation runs on average for each instance). The parameter settings used to generate instances are summarized in table 6.

Table 6: Parameter settings for the instances

Control parameter	Values
No. of activities (excluding the dummy activities)	10
No. of resource types	1
Resource availability	10 (also 15 and 20 in experiment 3)
Mean work content ( $\mu_i$ )	$U[10,50]$
Standard deviation of work content ( $\delta_i$ )	$U[1,5]$
Work content	$N(\mu_i, \delta_i)$
Instability weights for the non-dummy activities	$P(w_i = q) = (21 - 2q)\%, q \in \{1, 2, \dots, 10\}$
Instability weight for the dummy end activity	38
Instability weight for the dummy start activity	0
Due date	$s_n^{\min} \times 1.2$

## 4 Computational results

In this section, we obtain the computational results and evaluate the performance of roadrunner scheduling and railway scheduling according to the three experiments that are mentioned in section 3.5.

### 4.1 The influence of feeding buffer sizes

In this section, we investigate the impact of the use of feeding buffers combined with the roadrunner scheduling policy and the railway scheduling policy for each schedule and try to find an indication of the optimal feeding buffer size in terms of percentage of the non-critical chain length. Fig. 4 to 7 represent the results for the two scheduling policies, where an 'S' indicates the roadrunner scheduling policy, an 'R' indicates the railway scheduling policy and the numbers 0 to 5 represent feeding buffer sizes of 0 to 50%. For each combination of scheduling policy and feeding buffer size, the minimal, average and maximal result over all 100 instances is shown.

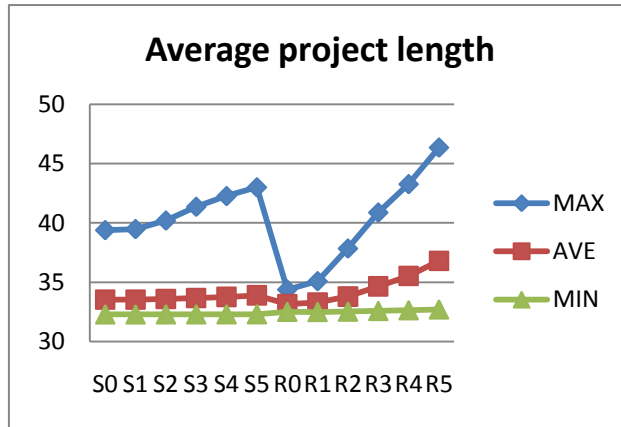


Fig. 4 APL of different feeding buffer sizes with roadrunner scheduling and railway scheduling

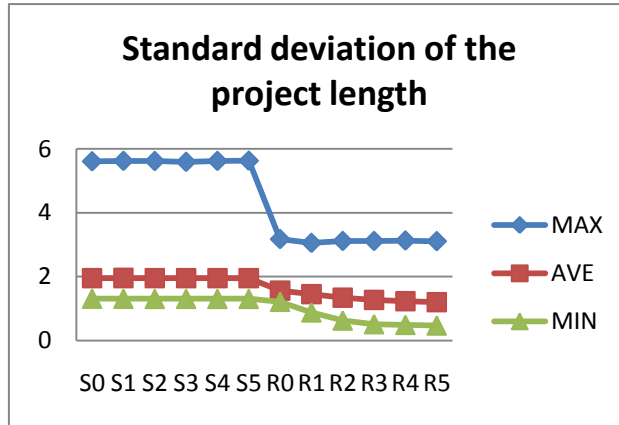


Fig. 5 *SDPL* of different feeding buffer sizes with roadrunner scheduling and railway scheduling

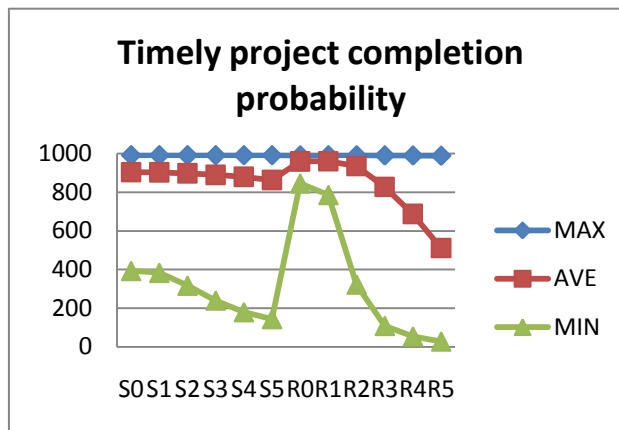


Fig. 6 *TPCP* of different feeding buffer sizes with roadrunner scheduling and railway scheduling

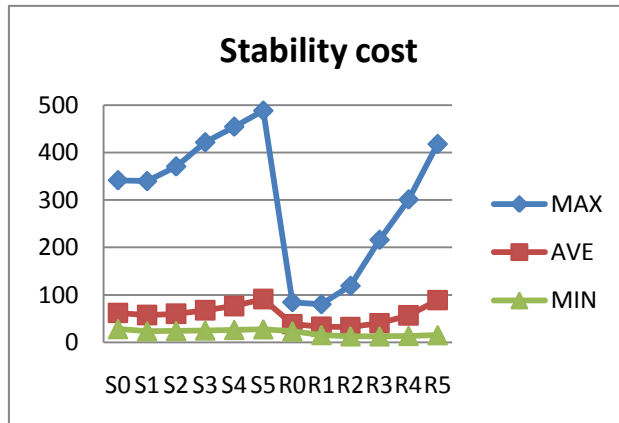


Fig. 7 *SC* of different feeding buffer sizes with roadrunner scheduling and railway scheduling

We can see from Fig. 4 to 7 that (perhaps surprisingly) execution policies R0 and R1 perform better than any other execution policy, except for the standard deviation of the project length where every increase in the feeding buffer size for the railway scheduling policy shows a slight improvement.

For the roadrunner scheduling policy, the average performance of execution

policies S0 to S5 is relatively stable over all performance indicators. For the railway scheduling policy, however, the average performance of execution policies R0 to R1 is clearly better than that of execution policies R2 to R5 (except for the standard deviation of the project length). For both roadrunner scheduling and railway scheduling policies, the stability cost decreases slightly at first and then increases with the increase of feeding buffer size.

We can draw the conclusion that for this project environment a railway scheduling policy combined with relatively small feeding buffers (0% or 10% of the non-critical chain length) can give better performance than the other options considered and this not only with respect to stability (as expected), but also with respect to the average project length and the timely project completion probability. The main reason for this result seems to be the fact that a project manager will typically select those mode combinations and that baseline schedule that result in a short project makespan. However, the resulting resource profile is so condense that one should stick as much as possible to this schedule and thus not schedule activities earlier when their predecessors have finished, possibly resulting in breaking up the critical chain.

#### **4.2 Different priority lists**

In this section, we will take a look at the simulation behavior of the roadrunner scheduling and the railway scheduling policy when combined with the eight different priority lists that were mentioned in Section 3.5.2. Fig. 8 to 11 represent the results for the two scheduling policies, where a 'P' indicates the roadrunner scheduling policy, a 'Q' indicates the railway scheduling policy and the numbers 1 to 8 represent the eight priority lists. For each combination of a scheduling policy and a priority list, the minimal, average and maximal result over all 100 instances is shown.

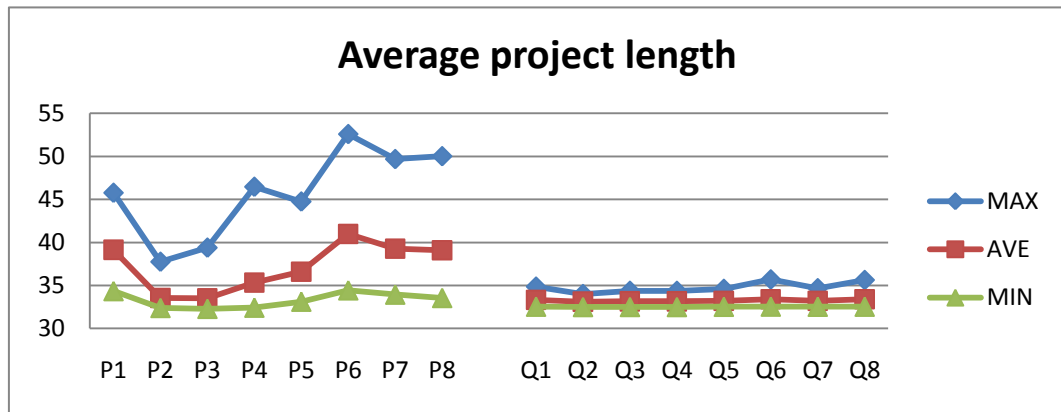


Fig. 8 APL of different priority lists according to roadrunner scheduling and railway scheduling

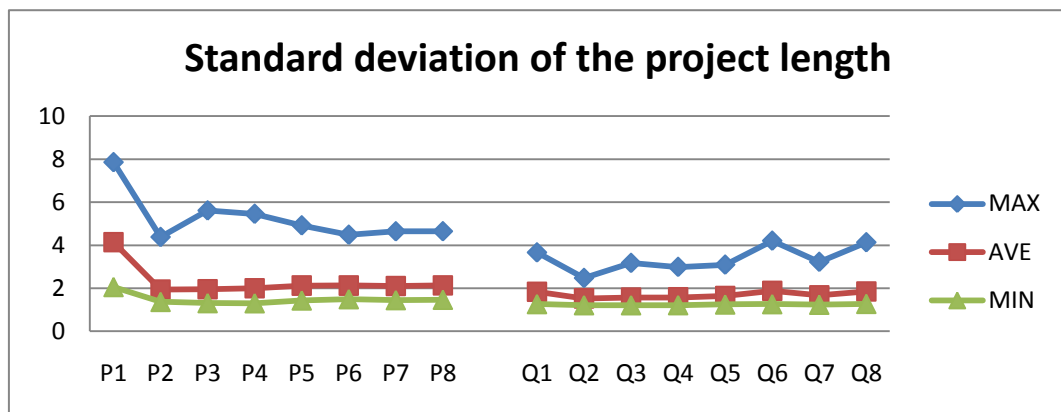


Fig. 9 SDPL of different priority lists according to roadrunner scheduling and railway scheduling

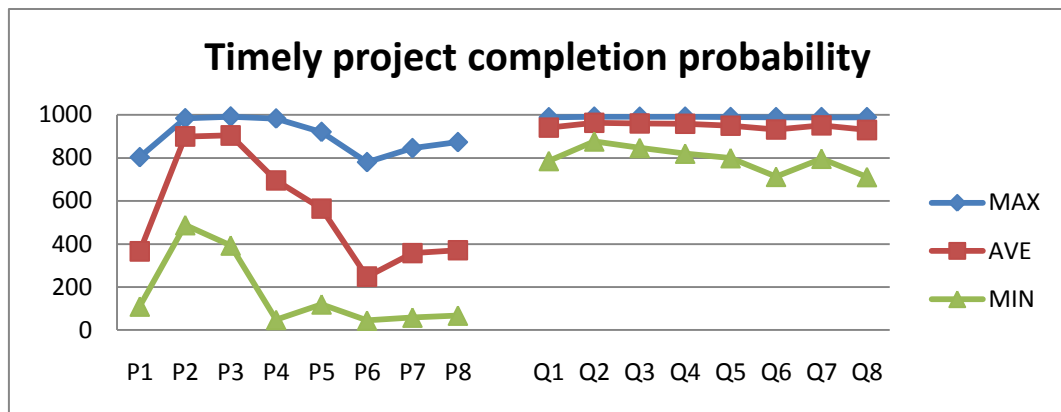


Fig. 10 TPCP of different priority lists according to roadrunner scheduling and railway scheduling



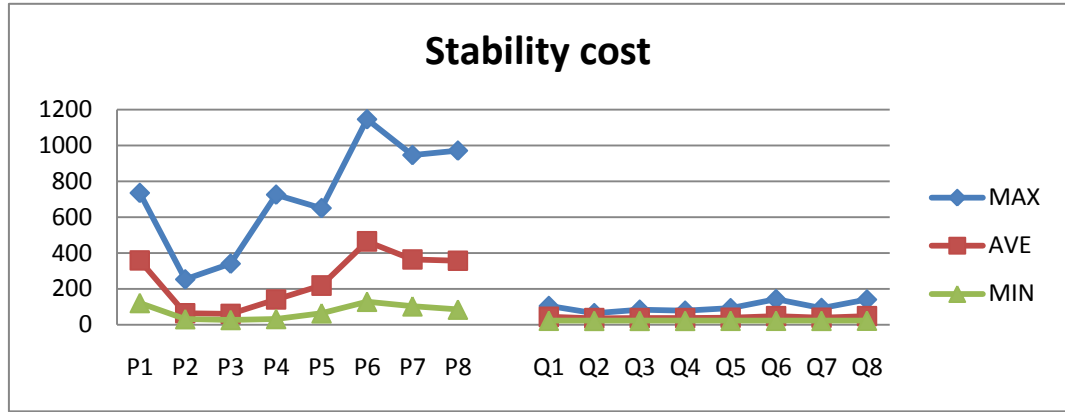


Fig. 11 SC of different priority lists according to roadrunner scheduling and railway scheduling

It is immediately clear from figures 8 to 11 that for all four performance indicators the sensitivity of the scheduling policy to the priority list is much smaller for the railway scheduling policy than for the roadrunner scheduling policy. Among the priority lists that are combined with the railway scheduling policy, the starting time (Q2) and first critical chain (Q3) priority lists have a slight edge on the other six priority lists.

Detailed analysis of the results obtained for the roadrunner scheduling policy indicated that the starting time (P2) and first critical chain (P3) priority lists clearly perform best when considering all four performance indicators. Comparing the first critical chain priority list (P3) with the second critical chain priority list (P4), it is obvious that P4 performs worse than P3, which means that adding the precedence constraints of the activity on the critical chain and forcing the critical chain to remain a chain can result in a smaller average project length, a higher timely project completion probability and a smaller stability cost. The random priority list P1 which creates random priority lists every time indeed has the highest standard deviation of the project length, but it performs better than P6, P7 and P8 for the other performance indicators.

We can conclude from this experiment that the railway scheduling policy using the starting time (Q2) and first critical chain (Q3) priority lists performs best for the problem type that is considered.

#### 4.3 Impact of the resource availability

In this section, we investigate the impact of the resource availability. The

computational results for the three resource availabilities for the first experiment are shown in Fig. 12 to 15 and for the second experiment in Fig. 16 to 19, where the three lines represent the average results of the 100 test instances for the three resource availabilities.

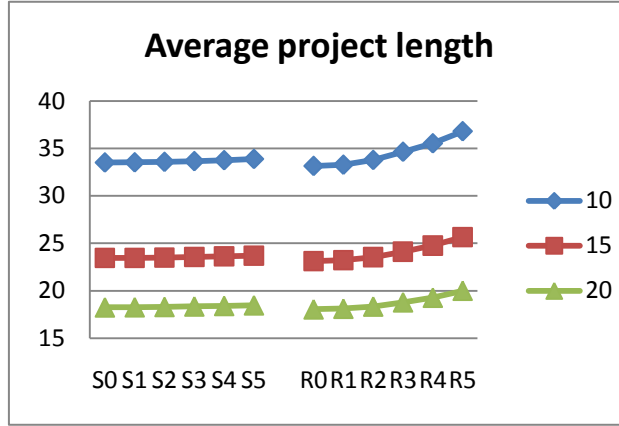


Fig. 12 *APL* for the three resource availabilities for the first experiment

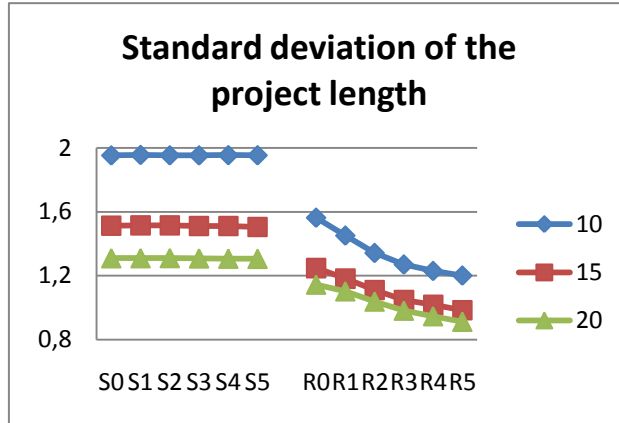


Fig. 13 *SDPL* for the three resource availabilities for the first experiment

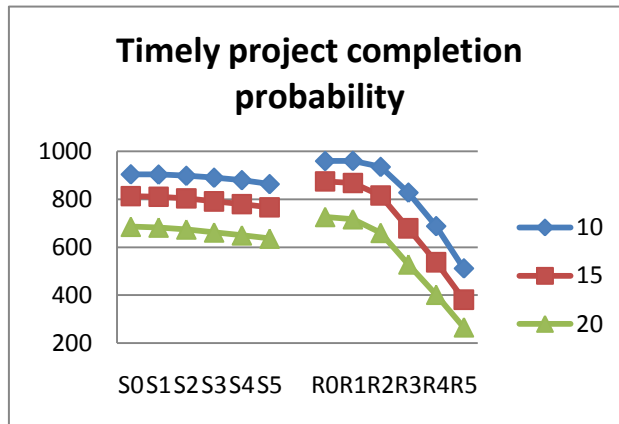


Fig. 14 *TPCP* for the three resource availabilities for the first experiment

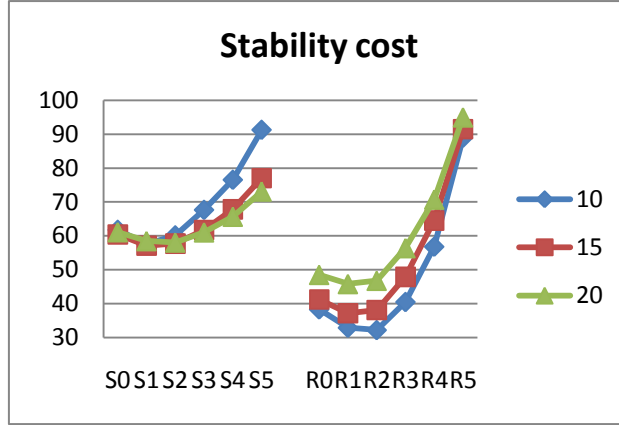


Fig. 15 SC for the three resource availabilities for the first experiment

In Fig. 12 to 14, the average curves show that the higher the resource availability, the smaller the average project length, the smaller the standard deviation of the project length and the smaller the timely project completion probability with both the roadrunner scheduling and the railway scheduling policies. One should remark that the optimal project makespan differs with different resource availabilities and thus the due date for the three cases is different (the due date is set at 20% over the optimal project makespan). Additionally, the average values of R3 to R5 increase clearly for the average project length while the average results of R3 to R5 decrease drastically for both the standard deviation of the project length and the timely project completion probability. The average results of R0 and R1 are slightly smaller than the results of S0 and S1 for the average project length and the average results of R0 and R1 are slightly larger than the results of S0 and S1 for the timely project completion probability. In Fig. 15, with the resource availability increasing, the average stability costs decrease for the roadrunner scheduling policy while they increase for the railway scheduling policy. For all the resource availabilities, the average stability cost first decreases and then increases dramatically as the feeding buffer increases. This last effect is mainly the result of the increased probability that the project length does not meet the due date. Indeed, small feeding buffers increase the stability of the real activities, but from a certain feeding buffer size on this decrease in stability cost is offset by the increase in the stability cost that is caused by not meeting the due date anymore. This effect seems to play stronger when the railway scheduling policy is used.

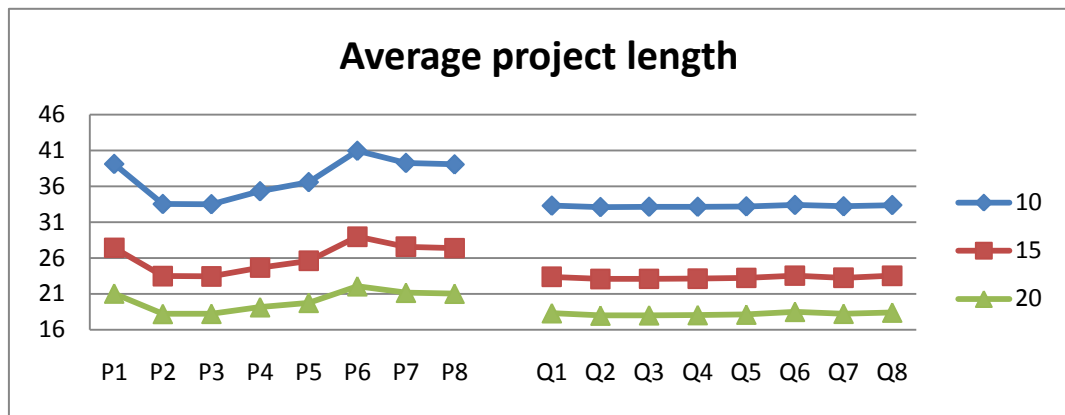


Fig. 16 APL for the three resource availabilities for the second experiment

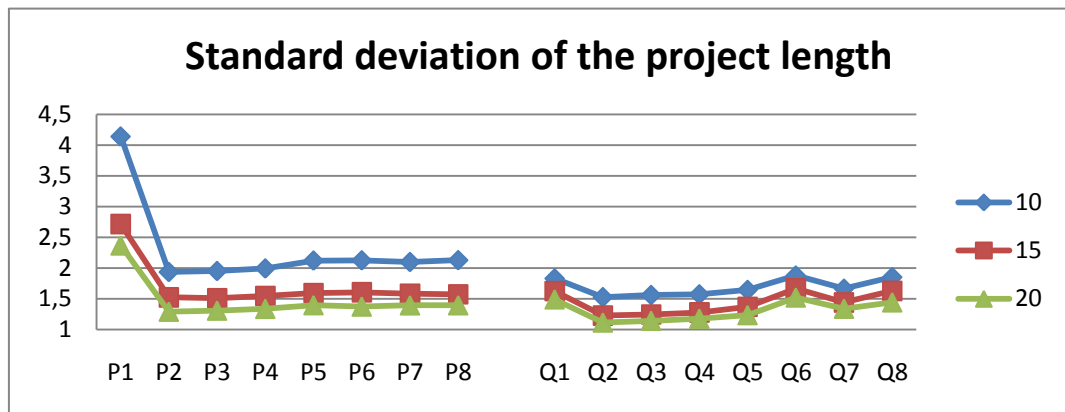


Fig. 17 SDPL for the three resource availabilities for the second experiment

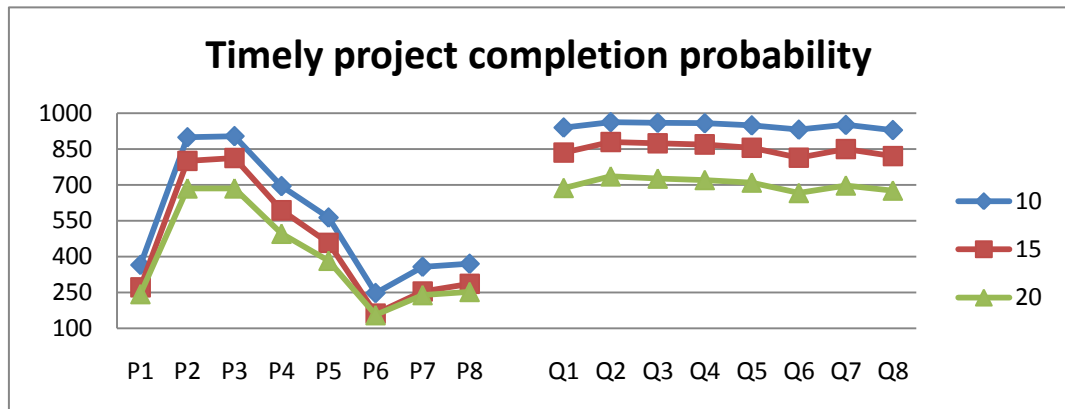


Fig. 18 TPCP for the three resource availabilities for the second experiment

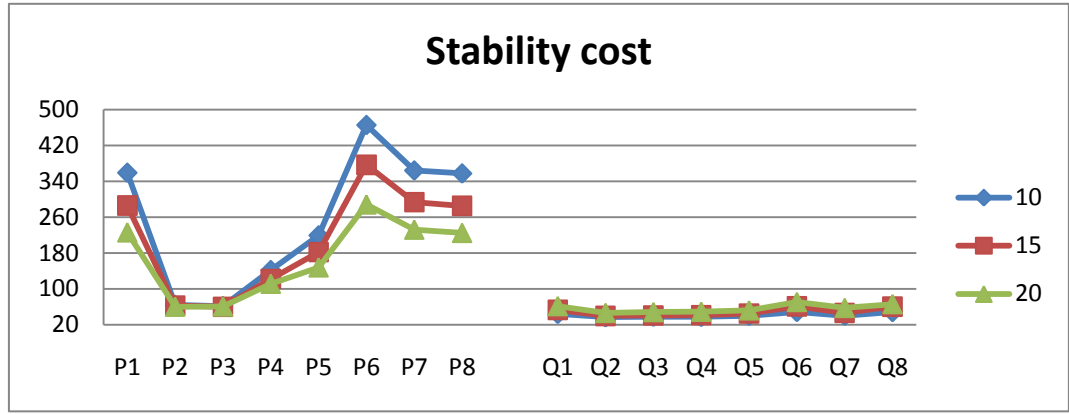


Fig. 19 SC for the three resource availabilities for the second experiment

Fig. 16 to 19 clearly indicate that the results with the railway scheduling policy are much better and much more stable than those based on the roadrunner scheduling policy no matter the resource availability. In Fig. 16 to 18, the average curves indicate that the higher the resource availability, the smaller the average project length, the smaller the standard deviation of the project length and the smaller the timely project completion probability (for different due dates) with both the roadrunner scheduling and the railway scheduling policies. In Fig. 19, with the resource availability increasing, the average stability cost decreases with the roadrunner scheduling policy while the average stability cost increases (very slightly) with the railway scheduling policy. Besides, we can also find that for both scheduling policies the starting time (P2 and Q2) and first critical chain (P3 and Q3) priority lists perform better than the others although this effect is much smaller for the railway scheduling policy.

## 5 Conclusion and topics for further research

The computational results in this paper indicate that for the very condense schedules that project managers might prefer in practice railway scheduling clearly performs better than roadrunner scheduling, not only for the stability cost and the standard deviation of the project length (as could be expected), but also for the average project length and the timely project completion probability. In a first computational experiment, we also found that in this project environment railway scheduling should typically be combined with relatively small feeding

buffers (0% to 10%). In the second computational experiment, it seems that a choice for the starting time and first critical chain priority lists results in the best outcomes when considering all four performance indicators. A third experiment indicated that the resource availability has a huge impact on the performance.

From the computational set-up, we have to point out that the computational results depend heavily on the optimal baseline schedule and the optimal rescheduled schedule. Furthermore, in this paper one optimal mode combination results in just one optimal schedule. Actually one optimal mode combination might have more than one optimal schedule which can have different effects on the final results. Therefore, how to generate stable optimal baseline schedules can be a topic for further research.

In this paper, we have enumerated all mode combinations in order to obtain those that result in a minimal project makespan. However, for somewhat larger projects this approach will no longer be feasible. Determining how to select the mode combinations which have the best performance when one is unable to enumerate all possible mode combinations could be another interesting topic for future research.

#### **Acknowledgements**

This research has been supported by the China Scholarship Council. We would like to acknowledge the China Scholarship Council for the financial support and the Research Center for Operations Management of the Katholieke Universiteit Leuven for providing a visiting research period to Wendi Tian.

#### **References**

- Alcaraz, J., Maroto, C., Ruiz, R. (2003). Solving the multi-mode resource constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Society*, 54(6), 614–626.
- Boctor, F.F. (1993). Heuristics for scheduling projects with resource restrictions and several resource-duration modes. *International Journal of Production Research*, 31(11), 2547–2558.
- Boctor, F.F. (1996a). Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, 34, 2335–2352.
- Boctor, F.F. (1996b). A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. *European Journal of Operational Research*, 90(2), 349–361.
- Bouleimen, K., Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode versions. *European Journal of Operational Research*, 149(2), 268–281.

- De, P., Dunne, E.J., Gosh, J.B., Wells, C.E. (1997). Complexity of the discrete time/cost problem for project networks. *Operations Research*, 45, 302-306.
- De Reyck, B., Demeulemeester, E., Herroelen, W. (1998). Local search methods for the discrete time/resource trade-off problem in project networks, *Naval Research Logistics Quarterly*, 45, 553-578.
- De Reyck, B. (1998). Scheduling projects with generalized precedence relations: Exact and heuristic procedures, Ph.D. Thesis, Department of Applied Economics, Katholieke Universiteit Leuven, Leuven, Belgium.
- Demeulemeester, E., Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource constrained project scheduling problem. *Management Science*, 38(12), 1803-1818.
- Demeulemeester, E., De Reyck, B., Herroelen, W. (2000) The discrete time/resource trade-off problem in project networks: A branch and bound approach, *IIE Transactions*, 32, 1059–1069.
- Demeulemeester, E., Herroelen, W. (2002). Project scheduling: A research handbook (pp. 473-530). Kluwer Academic Publishers
- Drexl, A., Grünewald, J. (1993). Non-preemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 25, 74–81.
- Elmaghraby, S.E. (2005). On the fallacy of averages in project risk management. *European Journal of Operational Research*, 165(2), 307-313.
- Fündeling, C.U., Trautmann, N. (2010). A priority-rule method for project scheduling with work-content constraints. *European Journal of Operational Research*, 203(3), 568-574.
- Goldratt, E.M. (1997). *Critical Chain*. New York: North River Press.
- Hartmann, S., Sprecher, A. (1996). A note on “hierarchical models for multi-project planning and scheduling”. *European Journal of Operational Research* 94(2), 377–383.
- Hartmann, S., Drexl, A. (1998). Project scheduling with multiple modes: A comparison of exact algorithms. *Networks*, 32, 283–297.
- Hartmann, S. (2001). Project scheduling with multiple modes: A genetic algorithm. *Annals of Operations Research*, 102, 111–135.
- Herroelen, W., Demeulemeester, E., De Reyck, B. (1999). A classification scheme for project scheduling. In Jan Weglarz (Ed.), *project scheduling: recent models, algorithm, and applications* (pp. 1–26). Kluwer Academic Publishers.
- Herroelen, W., Demeulemeester, E., De Reyck, B. (2000). On the paper “Resource-constrained project scheduling: Notation, classification, models, and methods” by Brucker et al., *European Journal of Operational Research*, 128(3), 221-230.
- Herroelen, W., Leus, R. (2001). On the merits and pitfalls of critical chain scheduling. *Journal of Operational Management*, 19, 559-577.
- Herroelen, W., Leus, R. (2004). Robust and reactive project scheduling: A review and classification of procedures. *International Journal of Production*, 42(8), 1599-1620.
- Jarboui, B., Damak, N., Siarry, P., Rebai, A. (2008). A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, 195(1), 299–308.
- Józefowska, J., Mika, M., Różycki, R., Waligóra, G., Weglarz, J. (2001). Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, 102, 137–155.
- Knotts, G., Dror, M., Hartman, B. (2000). Agent-based project scheduling. *IIE Transactions*, 32(5), 387–401.

Kolisch, R., Sprecher, A., Drexel, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, 41(10), 1693–1703.

Kolisch, R., Sprecher, A. (1996). PSPLIB – A project scheduling problem library. *European Journal of Operational Research*, 96(1), 205–216.

Kolisch, R., Drexel, A. (1997). Local search for nonpreemptive multi-mode resource constrained project scheduling. *IIE Transactions*, 29(11), 987–999.

Leach, L.P. (1999). Critical chain project management improves project performance. *Project Management Journal*, 30(2), 39–51.

Leach, L.P. (2000). *Critical chain project management*. Artech House.

Long, L., Ohsato, A. (2008). Fuzzy critical chain method for project scheduling under resource constraints and uncertainty. *International Journal of Project Management*, 26, 688–698.

Lova, A., Tormos, P., Barber, F. (2006). Multi-mode resource constrained project scheduling: Scheduling schemes, priority rules and mode selection rules. *Inteligencia Artificial*, 30, 69–86.

Lova, A., Tormos, P., Cervantes, M., Barber, F. (2009). An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics*, 117(2), 302–316.

Maroto, C., Tormos, P. (1994). Project management: An evaluation of software quality. *International Transactions in Operational Research*, 1(2), 209–221.

Mori, M., Tseng, C. (1997). A genetic algorithm for multi-mode resource constrained project scheduling problem. *European Journal of Operational Research*, 100(1), 134–141.

Newbold, R.C. (1998). *Project management in the fast lane: Applying the theory of constraints*. St. Lucie Press.

Nonobe, K., Ibaraki, T. (2001). Formulation and tabu search algorithm for the resource constrained project scheduling problem (RCPSP). Technical Report, Kyoto University.

Özdamar, L., Ulusoy, G. (1994). A local constraint based analysis approach to project scheduling under general resource constraints. *European Journal of Operational Research*, 79(2), 287–298.

Özdamar, L. (1999). A genetic algorithm approach to a general category project scheduling problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 29(1), 44–59.

Patterson, J., Slowinski, R., Talbot, F., Weglarz, J. (1989). An algorithm for a general class of precedence and resource constrained scheduling problem. In Slowinski, R. and Weglarz, J. (Ed.), *Advances in Project Scheduling* (pp. 3–28). Amsterdam: Elsevier.

Ranjbar, M., Kianfar, F. (2007) Solving the discrete time/resource trade-off problem with genetic algorithms. *Applied Mathematics and Computation*, 191(2), 451–456.

Ranjbar, M., De Reyck, B., Kianfar, F. (2009). A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. *European Journal of Operational Research*, 193(1), 35–48.

Slowinski, R. (1980). Two approaches to problems of resource allocation among project activities – A comparative study. *Journal of the Operational Research Society*, 31(8), 711–723.

Slowinski, R., Soniewicki, B., Weglarz, J. (1994). DSS for multi objective project scheduling. *European Journal of Operational Research*, 79(2), 220–229.

Speranza, M., Vercellis, C. (1993). Hierarchical models for multi-project planning and scheduling. *European Journal of Operational Research*, 64, 312–325.

Sprecher, A. (1994). Resource-constrained project scheduling: Exact methods for the multi-mode case. *Lecture Notes in Economics and Mathematical Systems*, vol. 409, Berlin: Springer.



- Sprecher, A., Hartmann, S., Drexl, A. (1997). An exact algorithm for the project scheduling with multiple modes. *OR Spektrum* 19, 195–203.
- Sprecher, A., Drexl, A. (1998). Multi-mode resource-constrained project scheduling problems by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, 107(2), 431–450.
- Sung, C.S., Lim, S.K. (1997). A scheduling procedure for a general class of resource constrained projects. *Computers and Industrial Engineering*, 32, 9-17.
- Talbot, F. (1982). Resource-constrained project scheduling with time/resource tradeoffs: The nonpreemptive case. *Management Science*, 28(10), 1197–1210.
- Tukel, O.I., Rom, W. O., Eksioglu, S.D. (2006). An investigation of buffer sizing techniques in critical chain scheduling. *European Journal of Operational Research*, 172(2), 401-416.
- Umble, M., Umble, E. (2000). Manage your projects for success: An application of the theory of constraints. *Production and Inventory Management Journal*, 41(2), 27.
- Vanhoucke, M., Debels, D. (2008). The impact of various activity assumptions on the lead time and resource utilization of resource-constrained projects. *Computers and Industrial Engineering*, 54(1), 140–154.
- Van de Vonder, S., Demeulemeester, E., Herroelen, W., Leus, R. (2005). The use of buffers in project management: The trade-off between stability and makespan. *International Journal of Production Economics*. 97, 227-240.
- Van de Vonder, S., Demeulemeester, E., Herroelen, W. (2008). Proactive heuristic procedures for robust project scheduling: An experiment analysis. *European Journal of Operational Research*, 189, 723-733.
- Van Peteghem, V., Vanhoucke, M. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource constrained project scheduling problem. *European Journal of Operational Research*, 201(2), 409-418.
- Zhang, H., Tam, C.M., Li, H. (2006). Multimode project scheduling based on particle swarm optimization. *Computer Aided Civil and Infrastructure Engineering*, 21(2), 93–103.
- Zhu, G., Bard, J.F., Yu, G. (2006). A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *Journal on Computing*, 18(3), 377–390.